

**Arrays**

**Test grading**

The objective of this program is to gain experience in using arrays. This program is supposed to generate a numerical score and a letter grade for students based on their answers for an exam. Each question in the exam is of True/False type. There are exactly 25 questions on the exam. An exam answer key is provided in the input data. The student grade for the exam is calculated based on the answer key. You are supposed to use arrays to store the answer key and the student responses.

**Sample input data:**

There will be two input files. The first input file contains the answer key and is called AnswerKey.txt as shown below.

```

Answer Key
-----
TTTTTTTTFTFFFTFTFTFTFFFT
    
```

The first two lines are just header lines and can be safely discarded.

The second input file is named StudentAnswers.txt, as indicated below.

SID	Student Response	Name
053688709	FFFFFFFFFTFTFTFTFTTT F	Bugs Bunny
207655304	TTTTTTTTFTFFFTFTFTFTFFFT	Donald Duck
137119833	F TTTTTFTFFFTFTFTFTFTTF	Mickey Mouse
943640504	FTTFTFTFTFTFTFTFTFTFT	Roadrunner
279102885	FFFFFFFFFTFTFTFTFTFFFT	Minnie Mouse

The following rules apply for the student input data:

1. The first two lines are header lines and can be safely ignored.
2. Each line contains a student ID followed by the character '|', the student responses, followed by the character '|' and the student name. An example of this is given below.  
123456789|TFTFTFT FFTTTTTFTFFTTFF|Yosemite Sam
3. There could be an empty space between the T and F characters indicating that the student did not answer the question. In the above example, the student did not answer question 9.
4. The number of students in the input file is not fixed. Assume that there will be at least one student line.
5. Each correct answer is worth 4 points, each incorrect answer is worth -1 points and an unanswered question is worth 0 points. Your program should compare the student responses to the key provided and generate an aggregate score for the exam.

**Output**

You are supposed to print out the name of the student, the id and the test score for each student based on their responses. After that, each line should contain the student ID followed by the test score and the test grade. The test grade is a letter grade based on the following scale:

Range of scores	Letter Grade
93 - 100	A
90 - 92	A-
87 - 89	B+
83 - 86	B
80 - 82	B-
77 - 79	C+
73 - 76	C
70 - 72	C-
67 - 69	D+
63 - 66	D
60 - 62	D-
0 - 59	F

The average score for the entire class should also be printed out.

## Sample output

Here is a sample output file, named `scores.txt`, which corresponds to the input data given above:

```

Programmer: Mir Farooq Ali
CS 1044 Spring 2004 Project 7

=====
Name           ID           Score    Letter Grade
=====
Bugs Bunny     053688709     62       D-
Donald Duck    207655304     95       A
Mickey Mouse   137119833     76       C
Roadrunner     943640504     71       C-
Minnie Mouse   279102885     95       A
=====
Average Score = 79.80
=====

```

The first two lines identify the programmer and project, as usual. Then there is a blank line, followed by three header lines. This is followed by the name, the ID, test score and letter grade for each student. This is followed by another trailing line, the average score, and finally, a trailing line.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

Note that you must insert blank lines and divider lines as shown.

## Program Requirements

You have to follow the requirements specified below. Failure to do so will result in deductions.

1. You are required to store the answer key in an array of characters.
2. Each student's responses should also be stored in an array of characters.
3. There should be at least one input function.
4. There should be at least one output function.
- 5.

## Evaluation:

Everything that was said in the specification for the earlier about testing still applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

### Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://eags.cs.vt.edu:8080/curator/>

### Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

**Failure to include this pledge in a submission is a violation of the Honor Code.**